TACO2+ over Combat Net Radio (MIL-STD-188-220A)

John C. C. White

June 1996

MITRE

Introduction

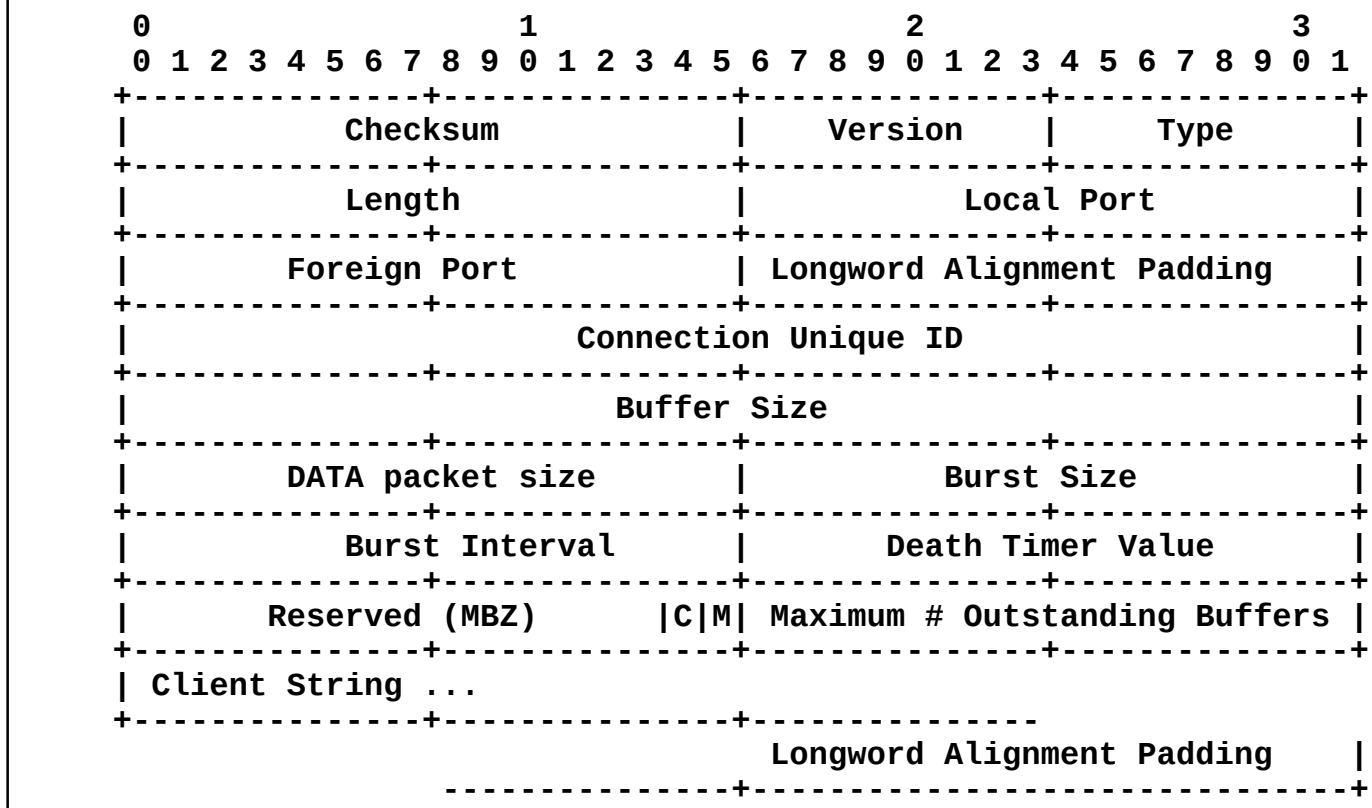
- There is a need for a bulk data transfer mechanism for the Combat Net Radio protocol suite
- NETBLT has been identified as a candidate for this role
- There has been considerable real-world experience with NETBLT, because it is the heart of TACO2, the tactical protocol component of the NITFS
- This briefing describes:
  - the basic operation of the NETBLT protocol
  - the results of initial tests conducted using that protocol over CNR at the Digital Integration Lab, Ft. Monmouth

Some Elements of NETBLT

- NETBLT is a bulk data transmission protocol
- Data is divided into “buffers”, which are further divided into (and transmitted as) packets
  - Buffers are acknowledged, packets are retransmitted
  - Multiple buffers may be outstanding concurrently
- Values of several variables are initially negotiated
- Operation is driven mainly from the receiver end
  - Reduces sensitivity to network delay
  - Retransmission timer is a critical element
- Burst size and burst interval provide a throttling mechanism
- There is a single control packet, which can hold multiple control messages

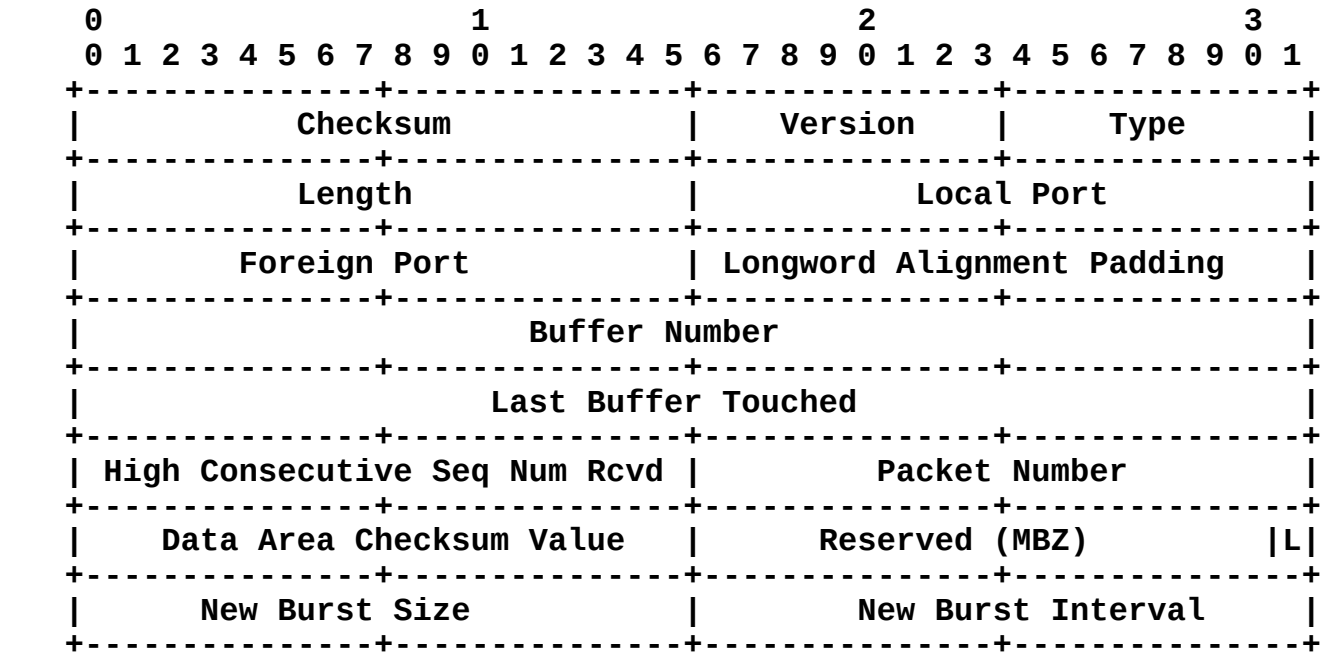
OPEN and RESPONSE Packet Format

5.2.6.1 OPEN (type 0) and RESPONSE (type 1):



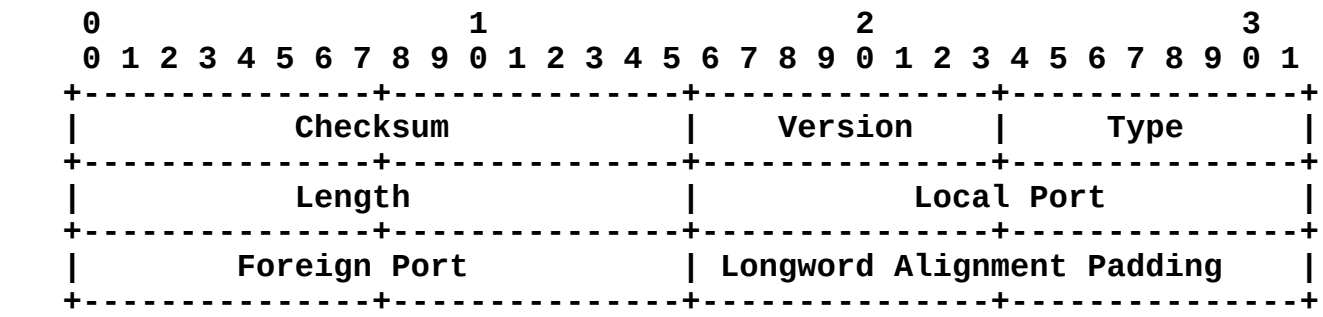
DATA Packet Format

5.2.6.4 DATA (type 5) and LDATA (type 6):



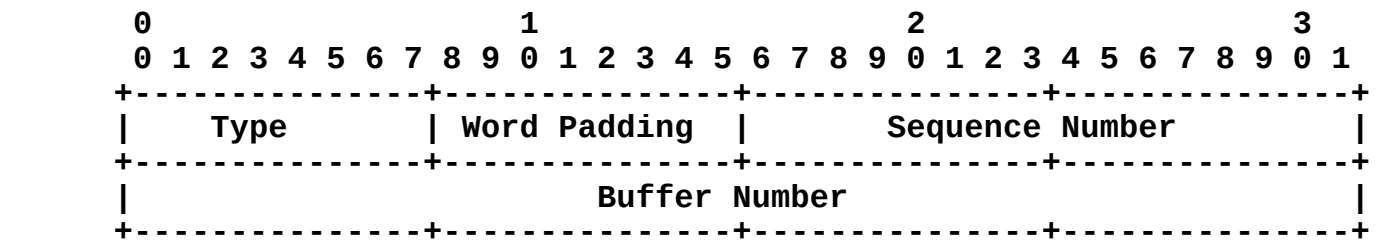
CONTROL Packet Header Format

5.2.6.6 CONTROL (type 8):

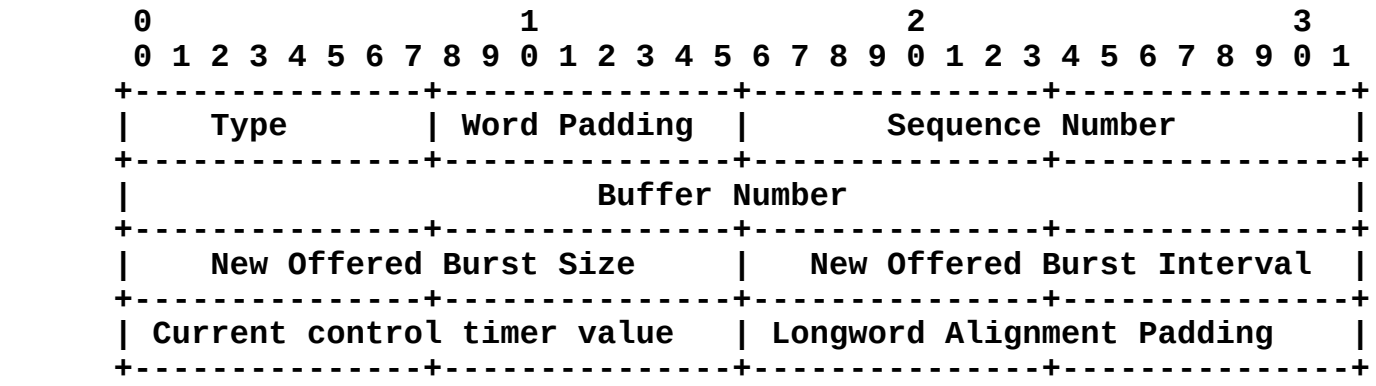


GO and OK Message Format

5.2.6.6.1 GO message (type 0):

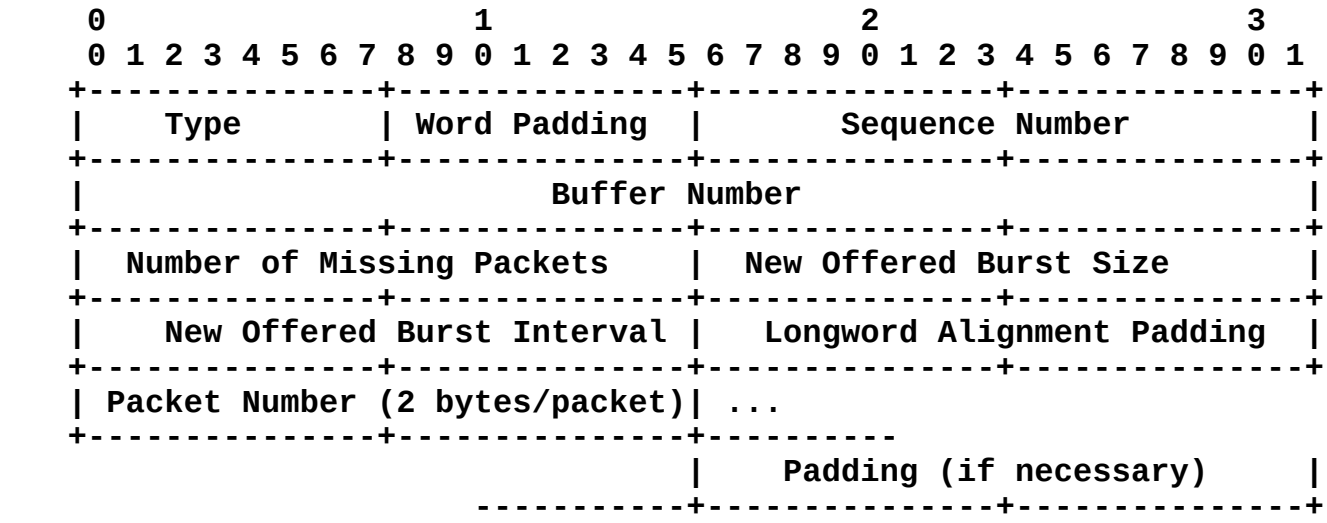


5.2.6.6.2 OK message (type 1):

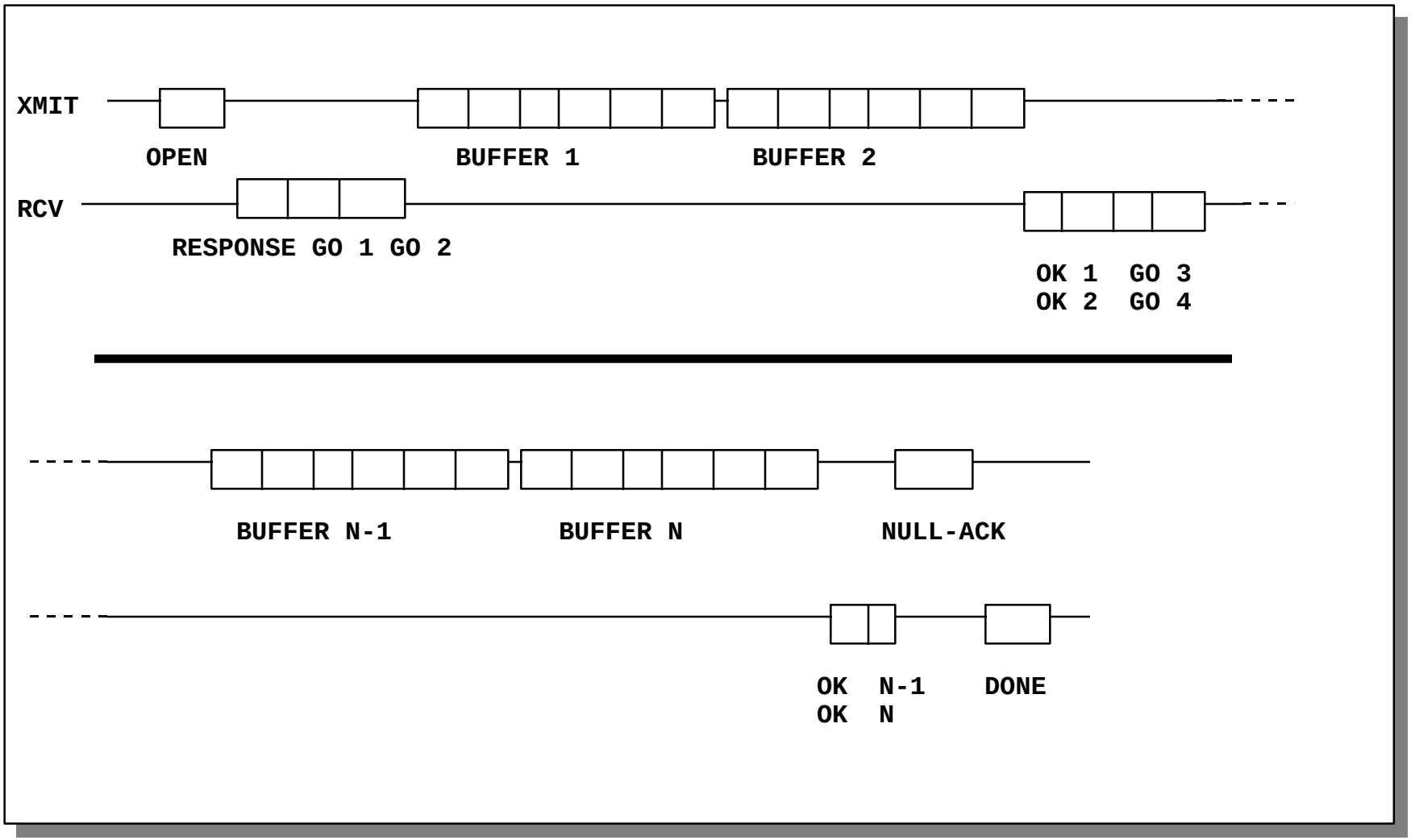


RESEND Message Format

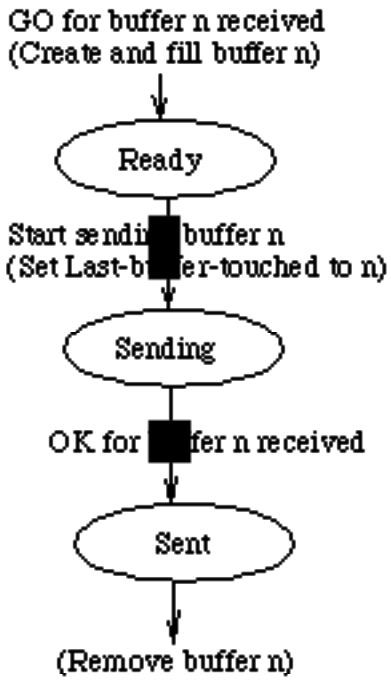
5.2.6.6.3 RESEND Message (type 2):



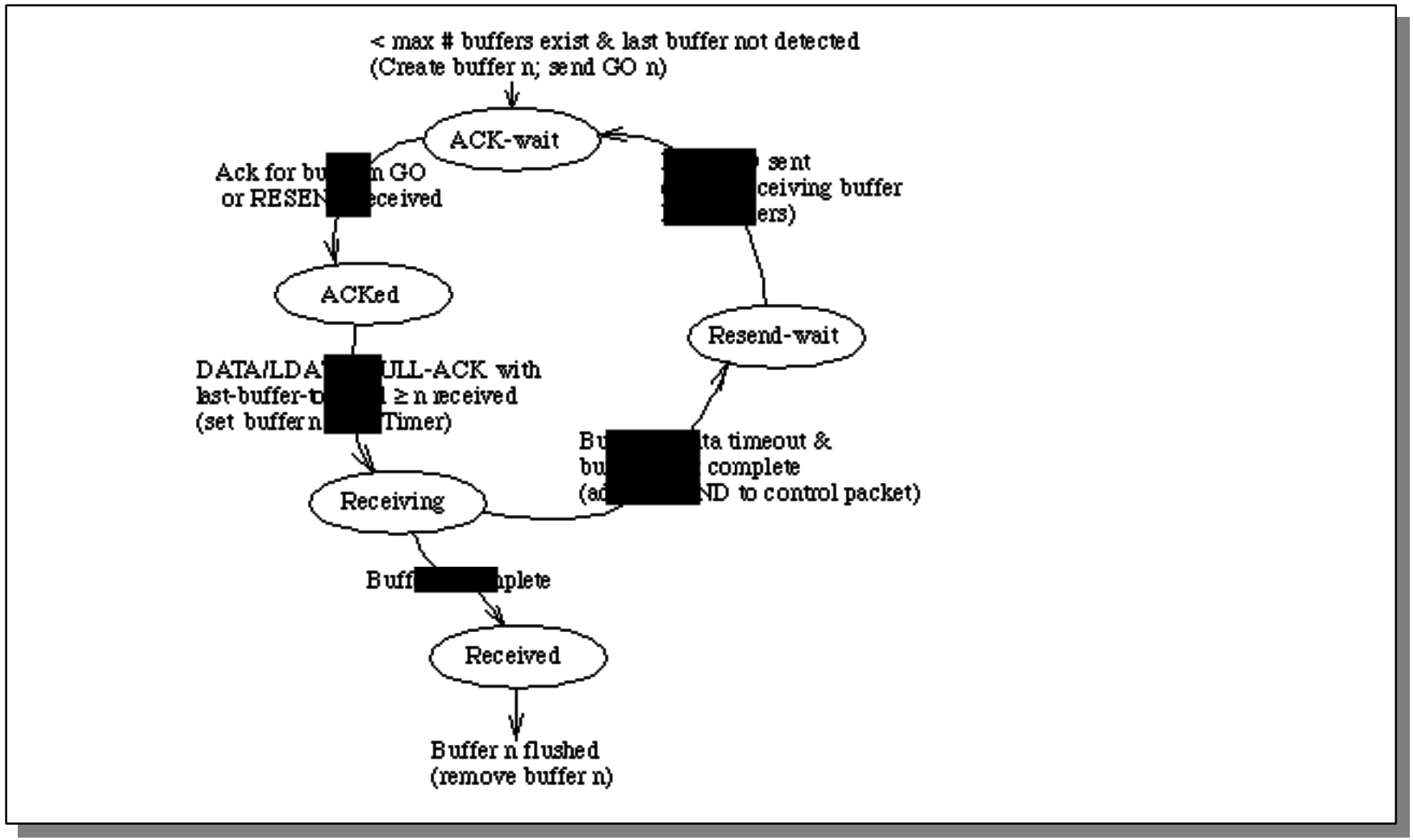
NETBLT Operation in “Half-duplex” Mode



Transmit Buffer State Diagram



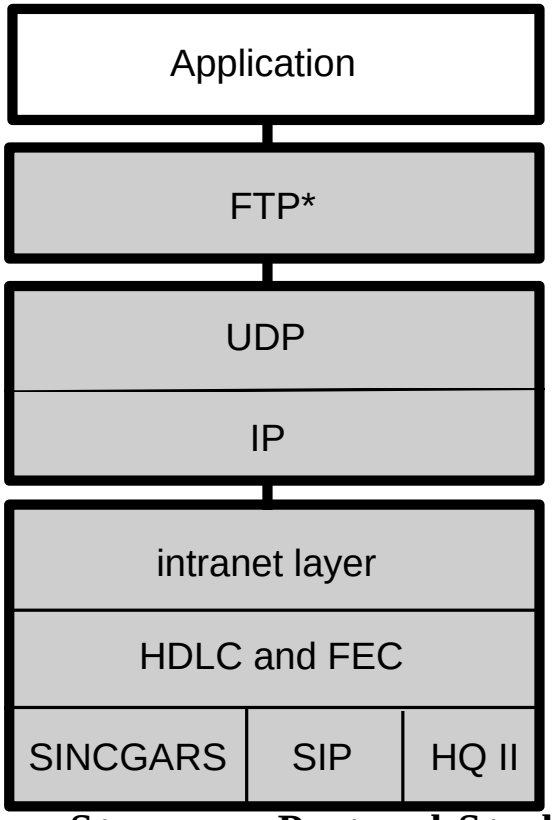
Receive Buffer State Diagram



Initial Testing

- Tests were conducted at the Digital Integration Lab at Ft. Monmouth NJ
- Network had just two nodes, hard-wired connectivity
- Normal MIL-STD-188-220A net access algorithms and FEC used, but no encryption
- Equipment was newly delivered to the DIL

Protocol Stack Tested



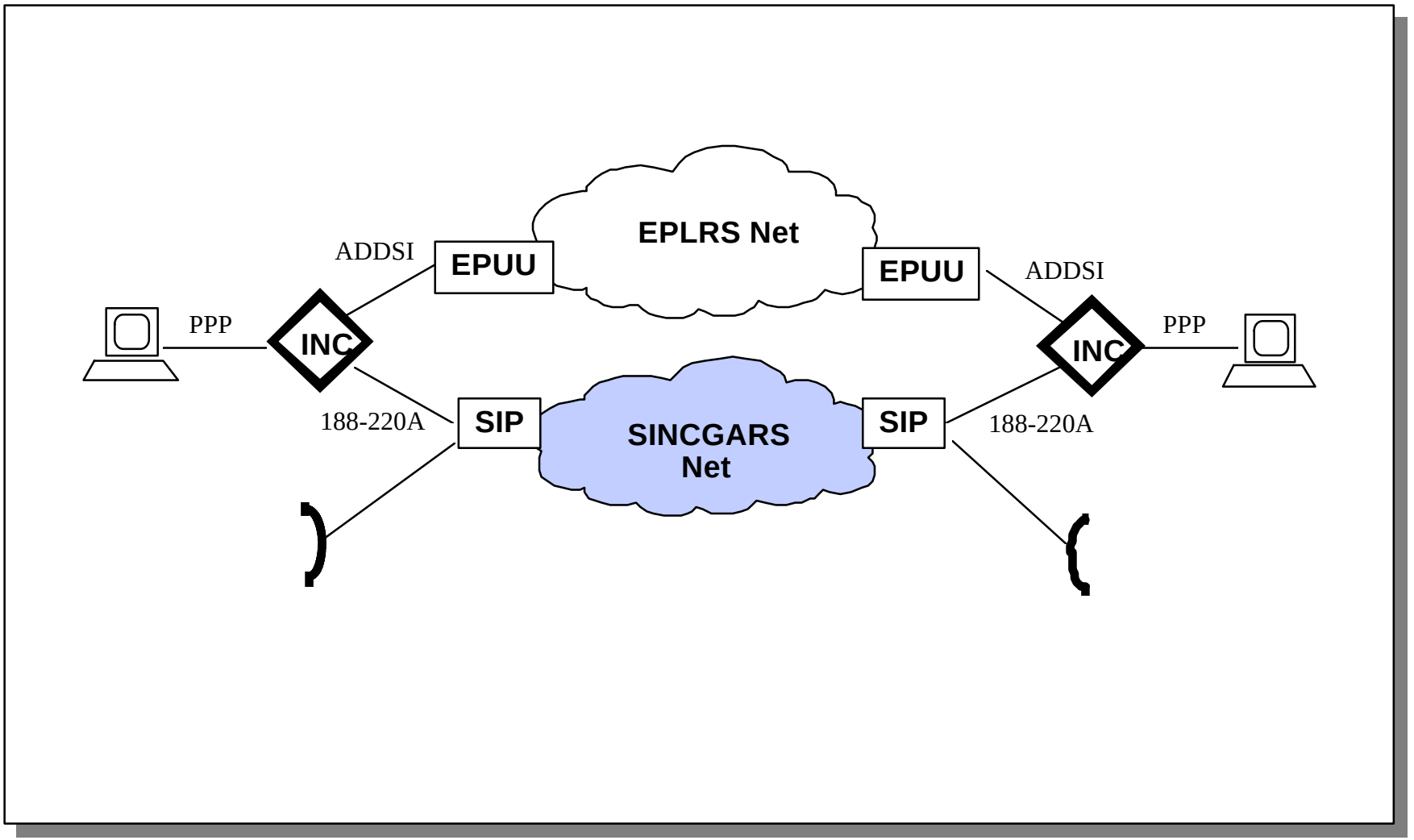
Derived from TACO2 Upper Layers

Internet Protocol Suite

MIL-STD-188-220A

Strawman Protocol Stack

Test Hardware Configuration



Initial Test Configuration

- At each node:
  - MS-DOS host computer
    - LSL.COM - Novell Netware Link Support Layer v2.05
    - SLIP_PPP.COM - Novell Asynchronous SLIP_PPP Driver v4.12
    - PPPPKT.COM - Packet Driver interface to ODI v0.60
    - PACKET.EXE - TACO2 Sample Software for MS-DOS v2.1beta
  - Internet Controller (INC)
  - SINGCARS SIP radio
  - radio handset
  - appropriate interconnecting cables

Test Observations

- Optimum packet size = 480 bytes (420 data, 60 header)
  - Allows three packets per transmission, usually
- Good buffer size = 4200 bytes, three buffers outstanding
- Half-duplex operation of NETBLT preferred over full-duplex
  - Fewer receiver to sender transmissions
  - Fewer radio collisions
- Presence or absence of UDP layer had negligible performance impact
- Local data rate > 19200 caused some PPP checksum errors
  - Processor and software dependent

Test Observations (cont)

- The INC uses the IP TOS field to control the operation of the link layer. We were able to operate the link in the following modes:
  - a. Type 1 datagram, no link-layer acknowledgement
  - b. Type 1 datagram, immediate ("coupled") transmission of a link-layer acknowledgement for each packet
  - c. Type 4 datagram, transmission of a link-layer acknowledgement for each transmission burst
- As expected, mode a was the most efficient in terms of throughput over a clean link, mode c was next most efficient, and mode b was the slowest, since it requires that the link be turned around for every packet

Test Observations (concluded)

- NETBLT Burst Interval set to fixed value worked best
  - INC delivers packets in bursts, which misled the current TACO2 packet timing algorithm
  - Improved packet timing algorithm will probably combine information from NETBLT Burst Interval and measured values with variance
- Brief network disturbances (generated by keying microphone or passing traffic across the second INC ports) had little effect
- Longer disturbances caused some retransmissions if packets were held up for too long

Test Setup

- Used three test files:
  - SMALL - 1000 bytes
  - MEDIUM - 10000 bytes
  - LARGE - 50000 bytes
- The tests were run with these initial parameters:
  - NETBLT packet size = 420 bytes
  - NETBLT buffer size = 4200 bytes
  - # of NETBLT buffers = 3
  - Burst interval = 10 seconds
  - Burst size = 10 packets
  - PPP rate = 38400
  - IP TOS = 0x10 (Routine precedence, D bit set = low delay)

Test Results

File	Time (sec)	Rate (bps)
SMALL	7.52	1063
	7.63	1048
	8.73	916
MEDIUM	39.50	2025
	32.52	2460
	43.50	1839
	44.55	1796

- In each of the last two runs, we observed that a control packet was lost and had to be retransmitted. We then reduced the PPP speed to 19200 bps and continued testing.

Test Results (cont)

- **LARGE**

160.52	2492
162.05	2468
161.23	2481

- **At this point, we changed the IP TOS field to 0x50 (Priority precedence, D bit set = low delay).**
- **LARGE**

157.11	2546
157.05	2547
157.71	2536

- **We then ran without the UDP layer.**
- **LARGE**

161.28	2480
155.08	2579
158.70	2520

Test Results (concluded)

- The UDP layer was restored, and the IP TOS field set to 0x00 (Routine precedence, every transmission burst is ACKed at the link layer).
- LARGE

213.88	1870
276.42	1447
264.50	1512

- One test was run with TOS set to 0x60 (Urgent precedence, every packet causes a coupled ACK). Transmission time was not recorded, but speed was about 960 bps for file LARGE.

Final Observation

- All 10000 bytes of MEDIUM are sent before the acknowledging GO is sent back from the receiver. Therefore, this portion of the data is being sent at the absolute maximum rate the link can achieve as configured.
- The difference between the fastest 1000-byte and the fastest 10000-byte transmission was 25 seconds
  - 9000 bytes * 8 bits per byte / 25 = 2880 bits per second with Routine precedence, D bit set for low delay
  - If we include the header bits for the 21 additional packets needed to send MEDIUM rather than SMALL, the maximum throughput is 10260 bytes * 8 bits per packet / 25 seconds = 3280 bps.
- This is the highest throughput that could be achieved across the link, as configured, by a "protocol" with no acknowledgements and no overhead whatsoever

Conclusion

- **The protocol stack under test achieved very close to the maximum possible throughput for a reliable transfer protocol across the network as configured.**
- **The primary focus for the next phase of testing will be on performance of a modified protocol stack (with improved packet timing algorithm) under less benign network conditions:**
  - **insertion of controlled amounts of additional, conflicting, network traffic**
  - **injection of link errors**
- **If possible we will operate with a more complex network, including EPLRS User Unit radios.**